HPC-AI-Report1

December 3, 2024

1 UM6P hackathon report (part 1)

1.1 TajTech

1.1.1 Mundiapolis

[ ]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

[3]:

```
import plotly.express as px

# File paths
file_no_slurm = "gemmtask1NoSlurm.csv"
file_slurm = "gemmtask1SLURM.csv"

# Load CSV data
df_no_slurm = pd.read_csv(file_no_slurm)
df_slurm = pd.read_csv(file_slurm)

# Add a label column to differentiate the data
df_no_slurm['Type'] = 'No SLURM'
df_slurm['Type'] = 'SLURM'

# Combine the two dataframes
df = pd.concat([df_no_slurm, df_slurm])

# Calculate MNK for better visualization
df['MNK'] = df['M'] * df['N'] * df['K']

# Create the scatter plot
fig = px.scatter(
    df,
    x='MNK',
    y='AverageTime(ms)',
    color='Type',
    hover_data={'M': True, 'N': True, 'K': True, 'PerformanceProportion(time/
 ↪MNK)': True},
    title="Comparison of Average Time vs. MNK",
    labels={"MNK": "M*N*K", "AverageTime(ms)": "Average Time (ms)"},
)

# Show the plot
fig.show()
```

[ ]:

```
# File path
file_path = "gemmtask.csv"

# Load CSV data
df = pd.read_csv(file_path)

# Calculate MNK for better visualization
df['MNK'] = df['M'] * df['N'] * df['K']

# Create a scatter plot for FLOPs vs. Performance(GFLOP/s)
fig = px.scatter(
    df,
    x='FLOPs',
    y='Performance(GFLOP/s)',
    size='MNK',  # Bubble size based on MNK
    color='MNK',  # Color based on MNK
    hover_data={'M': True, 'N': True, 'K': True, 'AverageTime(ms)': True},
    title="FLOPs vs. Performance (GFLOP/s)",
    labels={"FLOPs": "FLOPs", "Performance(GFLOP/s)": "Performance (GFLOP/s)"},
)

# Show the plot
fig.show()
```

[5]:

```
# File path
file_path = "gemmtaskSlurmFLOPS.csv"

# Load CSV data
df = pd.read_csv(file_path)

# Calculate MNK for better visualization
df['MNK'] = df['M'] * df['N'] * df['K']

# Create a scatter plot for FLOPs vs. Performance(GFLOP/s)
fig = px.scatter(
    df,
    x='FLOPs',
    y='Performance(GFLOP/s)',
    size='MNK',  # Bubble size based on MNK
    color='MNK',  # Color based on MNK
    hover_data={'M': True, 'N': True, 'K': True, 'AverageTime(ms)': True},
    title="FLOPs vs. Performance (GFLOP/s)",
    labels={"FLOPs": "FLOPs", "Performance(GFLOP/s)": "Performance (GFLOP/s)"},
)

# Show the plot
fig.show()
```

[6]:

```
# File path
file_path = "gemmtask1GPU.csv"

# Load CSV data
df_gpu = pd.read_csv(file_path)

# Calculate MNK for better visualization
df_gpu['MNK'] = df_gpu['M'] * df_gpu['N'] * df_gpu['K']

# Create a scatter plot for FLOPs vs. Performance(GFLOP/s)
fig = px.scatter(
    df_gpu,
    x='FLOPs',
    y='Performance(GFLOP/s)',
    size='MNK',  # Bubble size based on MNK
    color='MNK',  # Color based on MNK
    hover_data={'M': True, 'N': True, 'K': True, 'AverageTime(ms)': True},
    title="GPU: FLOPs vs. Performance (GFLOP/s)",
    labels={"FLOPs": "FLOPs", "Performance(GFLOP/s)": "Performance (GFLOP/s)"},
    log_x=True,  # Use a logarithmic scale for FLOPs if values vary greatly
)

# Show the plot
fig.show()
```

[7]:

```
# File path
file_path = "gemmtask1GPUEVENTS.csv"

# Load CSV data
df_gpu = pd.read_csv(file_path)

# Calculate MNK for better visualization
df_gpu['MNK'] = df_gpu['M'] * df_gpu['N'] * df_gpu['K']

# Create a scatter plot for FLOPs vs. Performance(GFLOP/s)
fig = px.scatter(
    df_gpu,
    x='FLOPs',
    y='Performance(GFLOP/s)',
    size='MNK',  # Bubble size based on MNK
    color='MNK',  # Color based on MNK
    hover_data={'M': True, 'N': True, 'K': True, 'AverageTime(ms)': True},
    title="GPU_Events: FLOPs vs. Performance (GFLOP/s)",
    labels={"FLOPs": "FLOPs", "Performance(GFLOP/s)": "Performance (GFLOP/s)"},
    log_x=True,  # Use a logarithmic scale for FLOPs if values vary greatly
)

# Show the plot
fig.show()
```

[9]:

```
# File path
file_path = "gemmtask1GPUvsCPU.csv"

# Load CSV data
df_gpu_cpu = pd.read_csv(file_path)

# Calculate MNK for better visualization
df_gpu_cpu['MNK'] = df_gpu_cpu['M'] * df_gpu_cpu['N'] * df_gpu_cpu['K']

# Create a scatter plot for FLOPs vs Performance for both CPU and GPU
fig = px.scatter(
    df_gpu_cpu.melt(
        id_vars=["FLOPs", "MNK"],
        value_vars=["PerformanceCPU(GFLOP/s)", "PerformanceGPU(GFLOP/s)"],
        var_name="Device",
        value_name="Performance (GFLOP/s)"
    ),
    x="FLOPs",
    y="Performance (GFLOP/s)",
    size="MNK",
    color="Device",
    hover_data={"MNK": True, "FLOPs": True},
    title="Performance Comparison: CPU vs GPU -Single Precision- ",
    labels={"FLOPs": "FLOPs", "Performance (GFLOP/s)": "Performance (GFLOP/s)"},
    log_x=True,  # Logarithmic scale for FLOPs
    log_y=True,  # Logarithmic scale for performance
)

# Show the plot
fig.show()
```

[11]:

```
# File path
file_path = "gemmtask1GPUvsCPUDOPUBLEPRECISION.csv"

# Load CSV data
df_gpu_cpu = pd.read_csv(file_path)
```

```
# Calculate MNK for better visualization
df_gpu_cpu['MNK'] = df_gpu_cpu['M'] * df_gpu_cpu['N'] * df_gpu_cpu['K']

# Create a scatter plot for FLOPs vs Performance for both CPU and GPU
fig = px.scatter(
    df_gpu_cpu.melt(
        id_vars=["FLOPs", "MNK"],
        value_vars=["PerformanceCPU(GFLOP/s)", "PerformanceGPU(GFLOP/s)"],
        var_name="Device",
        value_name="Performance (GFLOP/s)"
    ),
    x="FLOPs",
    y="Performance (GFLOP/s)",
    size="MNK",
    color="Device",
    hover_data={"MNK": True, "FLOPs": True},
    title="Performance Comparison: CPU vs GPU -Double Precision- ",
    labels={"FLOPs": "FLOPs", "Performance (GFLOP/s)": "Performance (GFLOP/s)"},
    log_x=True,  # Logarithmic scale for FLOPs
    log_y=True,  # Logarithmic scale for performance
)

# Show the plot
fig.show()
```

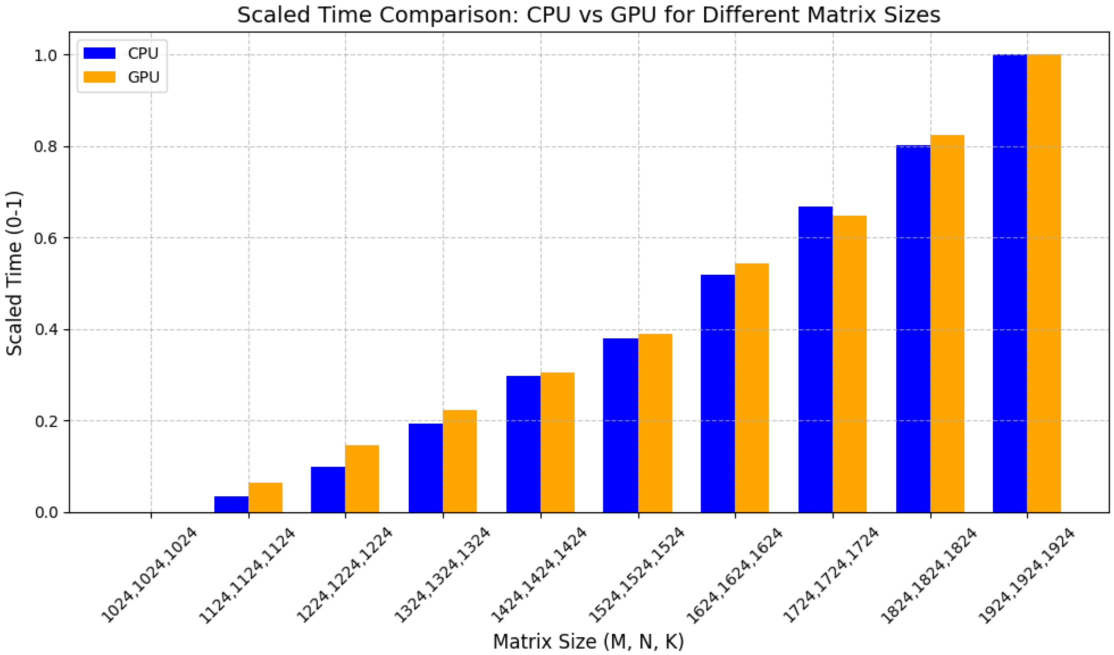
[29]:

```
import pandas as pd
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Load the CSV data
file_path = "gemmtask1GPUvsCPUbyTimeSameMNK.csv"
df = pd.read_csv(file_path)

# Add a new column to represent the matrix size (as a combination of M, N, K)
df['Matrix Size'] = df['M'].astype(str) + "x" + df['N'].astype(str) + "x" +␣
 ↪df['K'].astype(str)

# Rename columns for better readability
df = df.rename(columns={
    "AverageTimeCPU(ms)": "CPU Time",
    "AverageTimeGPU(ms)": "GPU Time",
})

# Create subplots for Average Time (CPU vs GPU)
fig = make_subplots(
    rows=1, cols=1,  # One row, one column for a single plot
    subplot_titles=("Average Time (CPU vs GPU)"),
)

# Bar plot for CPU Time
cpu_bar = go.Bar(
    x=df["Matrix Size"],
    y=df["CPU Time"],
    name="CPU",  # Name for the legend
    marker=dict(color='blue'),  # Blue color for CPU
)

# Bar plot for GPU Time
gpu_bar = go.Bar(
    x=df["Matrix Size"],
    y=df["GPU Time"],
    name="GPU",  # Name for the legend
    marker=dict(color='red'),  # Red color for GPU
)

# Add the traces to the plot
fig.add_trace(cpu_bar, row=1, col=1)
fig.add_trace(gpu_bar, row=1, col=1)

# Update layout for better visualization
fig.update_layout(
    title="Comparison of CPU vs GPU Time for Matrix Operations",
    xaxis_title="Matrix Size",
    yaxis_title="Average Time (ms)",
    xaxis_tickangle=-45,  # Rotate x-axis labels for better readability
    legend_title="Device",
    title_font_size=18,
    height=600,  # Set height for the plot
    barmode="group",  # Bars will be grouped next to each other for each matrix␣
 ↪size
)

# Show the plot
fig.show()
```

[33]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler

# Load the data from the CSV file
file_path = 'gemmtask1GPUvsCPUbyTimeSameMNK.csv'
data = pd.read_csv(file_path)
```

```
# Extract matrix sizes (M, N, K) and average times for CPU and GPU
matrix_sizes = data['M']
cpu_times = data['AverageTimeCPU(ms)']
gpu_times = data['AverageTimeGPU(ms)']

# Create a MinMaxScaler instance
scaler = MinMaxScaler()

# Reshape the data to 2D for scaling
cpu_scaled = scaler.fit_transform(cpu_times.values.reshape(-1, 1))
gpu_scaled = scaler.fit_transform(gpu_times.values.reshape(-1, 1))

# Plotting
bar_width = 0.35  # Width of the bars
index = range(len(matrix_sizes))

# Create the figure and axis
fig, ax = plt.subplots(figsize=(10, 6))

# Position the bars for CPU and GPU side by side
bar_cpu = ax.bar(index, cpu_scaled.flatten(), bar_width, label='CPU',␣
 ↪color='blue')
bar_gpu = ax.bar([i + bar_width for i in index], gpu_scaled.flatten(),␣
 ↪bar_width, label='GPU', color='orange')

# Set the labels for the x-axis
ax.set_xlabel('Matrix Size (M, N, K)', fontsize=12)
ax.set_ylabel('Scaled Time (0-1)', fontsize=12)

# Set the title of the plot
ax.set_title('Scaled Time Comparison: CPU vs GPU for Different Matrix Sizes',␣
 ↪fontsize=14)

# Set the X-axis ticks to represent the matrix size (aligned with the bars)
ax.set_xticks([i + bar_width / 2 for i in index])  # Shift X-axis for correct␣
 ↪alignment
ax.set_xticklabels([f'{m},{m},{m}' for m in matrix_sizes], rotation=45)

# Add legend
ax.legend()

# Show gridlines for better readability
ax.grid(True, linestyle='--', alpha=0.7)

# Show the plot
plt.tight_layout()
plt.show()
```

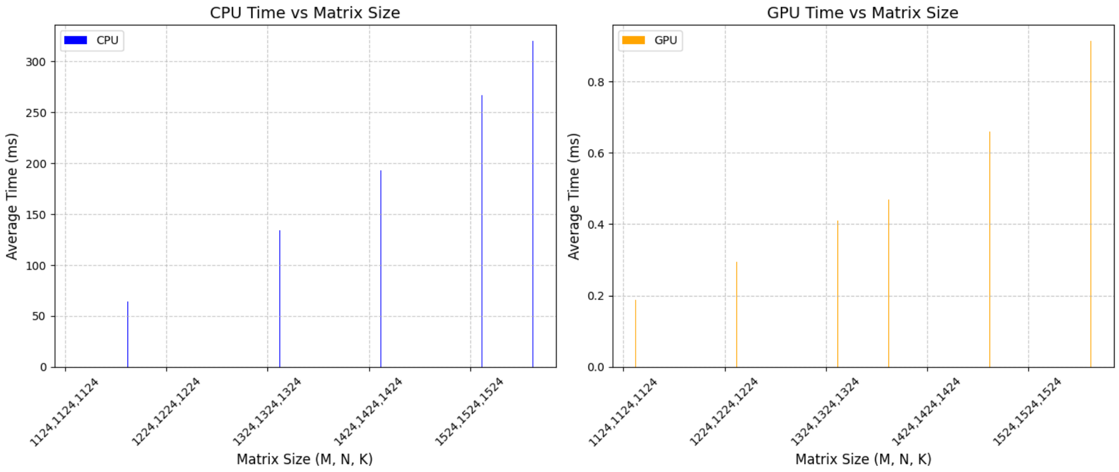


[34]:

```python
import pandas as pd
import matplotlib.pyplot as plt

# Load the data from the CSV file
file_path = 'gemmtask1GPUvsCPUbyTimeSameMNK.csv'
data = pd.read_csv(file_path)

# Extract matrix sizes (M, N, K) and average times for CPU and GPU
matrix_sizes = data['M']
cpu_times = data['AverageTimeCPU(ms)']
gpu_times = data['AverageTimeGPU(ms)']

# Create the figure and subplots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

# Plot CPU times in the first subplot
ax1.bar(matrix_sizes, cpu_times, color='blue', label='CPU')
ax1.set_xlabel('Matrix Size (M, N, K)', fontsize=12)
ax1.set_ylabel('Average Time (ms)', fontsize=12)
ax1.set_title('CPU Time vs Matrix Size', fontsize=14)
ax1.set_xticklabels([f'{m},{m},{m}' for m in matrix_sizes], rotation=45)
ax1.legend()
ax1.grid(True, linestyle='--', alpha=0.7)

# Plot GPU times in the second subplot
```

```python
ax2.bar(matrix_sizes, gpu_times, color='orange', label='GPU')
ax2.set_xlabel('Matrix Size (M, N, K)', fontsize=12)
ax2.set_ylabel('Average Time (ms)', fontsize=12)
ax2.set_title('GPU Time vs Matrix Size', fontsize=14)
ax2.set_xticklabels([f'{m},{m},{m}' for m in matrix_sizes], rotation=45)
ax2.legend()
ax2.grid(True, linestyle='--', alpha=0.7)

# Adjust layout for better spacing
plt.tight_layout()

# Show the plot
plt.show()
```

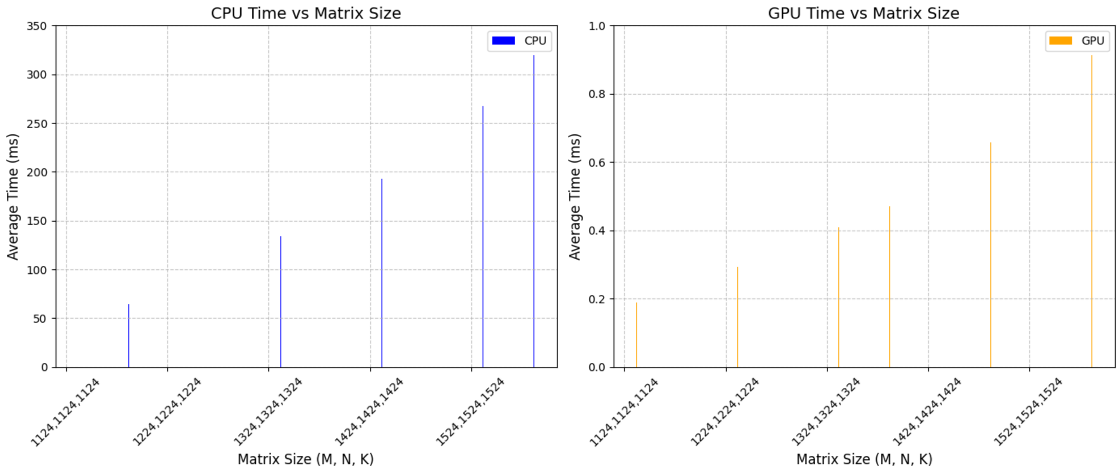
```
C:\Users\hp\AppData\Local\Temp\ipykernel_23072\2175332277.py:21: UserWarning:

set_ticklabels() should only be used with a fixed number of ticks, i.e. after
set_ticks() or using a FixedLocator.

C:\Users\hp\AppData\Local\Temp\ipykernel_23072\2175332277.py:30: UserWarning:

set_ticklabels() should only be used with a fixed number of ticks, i.e. after
set_ticks() or using a FixedLocator.
```



[35]:
```python
import pandas as pd
import matplotlib.pyplot as plt

# Load the data from the CSV file
file_path = 'gemmtask1GPUvsCPUbyTimeSameMNK.csv'
data = pd.read_csv(file_path)
```

```python
# Extract matrix sizes (M, N, K) and average times for CPU and GPU
matrix_sizes = data['M']
cpu_times = data['AverageTimeCPU(ms)']
gpu_times = data['AverageTimeGPU(ms)']

# Create the figure and subplots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

# Plot CPU times in the first subplot
ax1.bar(matrix_sizes, cpu_times, color='blue', label='CPU')
ax1.set_xlabel('Matrix Size (M, N, K)', fontsize=12)
ax1.set_ylabel('Average Time (ms)', fontsize=12)
ax1.set_title('CPU Time vs Matrix Size', fontsize=14)
ax1.set_xticklabels([f'{m},{m},{m}' for m in matrix_sizes], rotation=45)
ax1.set_ylim(0, 350)  # Adjust the y-axis to reflect the max CPU time (up to 350ms)
ax1.legend()
ax1.grid(True, linestyle='--', alpha=0.7)

# Plot GPU times in the second subplot
ax2.bar(matrix_sizes, gpu_times, color='orange', label='GPU')
ax2.set_xlabel('Matrix Size (M, N, K)', fontsize=12)
ax2.set_ylabel('Average Time (ms)', fontsize=12)
ax2.set_title('GPU Time vs Matrix Size', fontsize=14)
ax2.set_xticklabels([f'{m},{m},{m}' for m in matrix_sizes], rotation=45)
ax2.set_ylim(0, 1)  # Adjust the y-axis to reflect the max GPU time (up to 1ms)
ax2.legend()
ax2.grid(True, linestyle='--', alpha=0.7)

# Adjust layout for better spacing
plt.tight_layout()

# Show the plot
plt.show()
```

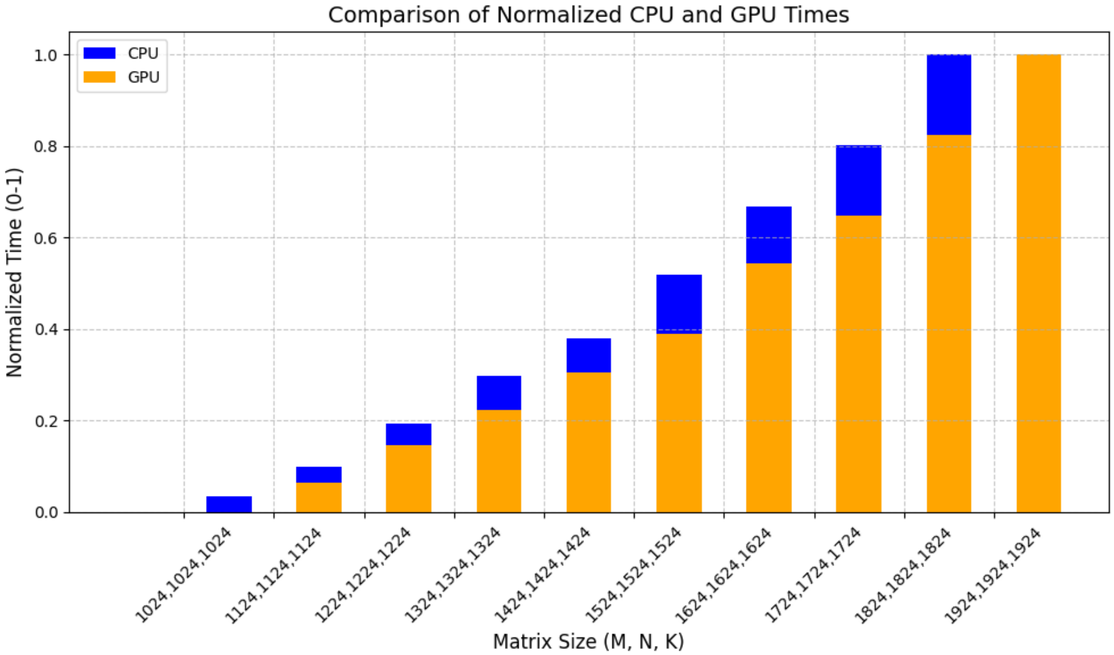
```
C:\Users\hp\AppData\Local\Temp\ipykernel_23072\2770043571.py:21: UserWarning:

set_ticklabels() should only be used with a fixed number of ticks, i.e. after
set_ticks() or using a FixedLocator.

C:\Users\hp\AppData\Local\Temp\ipykernel_23072\2770043571.py:31: UserWarning:

set_ticklabels() should only be used with a fixed number of ticks, i.e. after
set_ticks() or using a FixedLocator.
```



[36]:

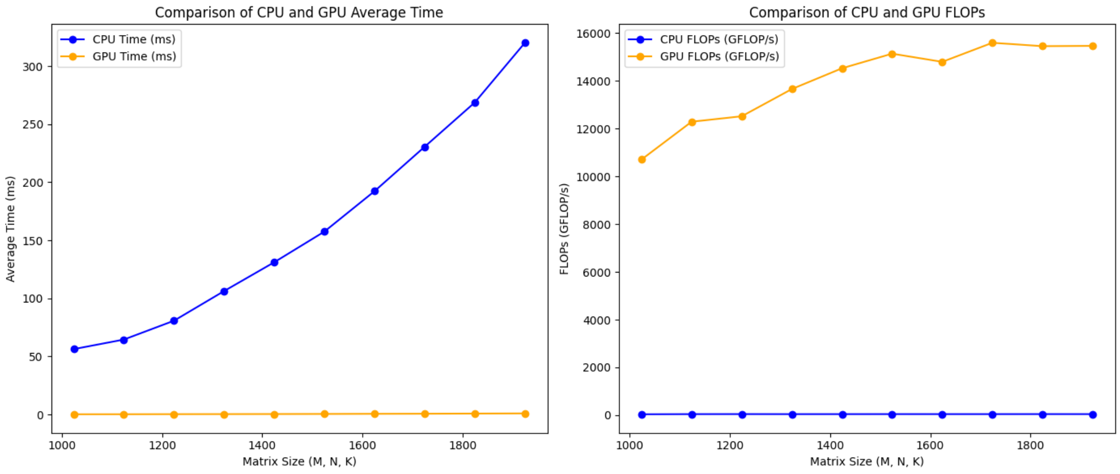
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Load the data from the CSV file
file_path = 'gemmtask1GPUvsCPUbyTimeSameMNK.csv'
data = pd.read_csv(file_path)

# Extract matrix sizes (M, N, K) and average times for CPU and GPU
matrix_sizes = data['M']
cpu_times = data['AverageTimeCPU(ms)']
gpu_times = data['AverageTimeGPU(ms)']

# Normalize CPU and GPU times (scale to 0-1 range)
cpu_times_normalized = (cpu_times - cpu_times.min()) / (cpu_times.max() -␣
 ↪cpu_times.min())
gpu_times_normalized = (gpu_times - gpu_times.min()) / (gpu_times.max() -␣
 ↪gpu_times.min())

# Create the figure and subplots
fig, ax = plt.subplots(figsize=(10, 6))

# Plot normalized CPU times
ax.bar(matrix_sizes - 50, cpu_times_normalized, width=50, color='blue',␣
 ↪label='CPU', align='center')

# Plot normalized GPU times
ax.bar(matrix_sizes + 50, gpu_times_normalized, width=50, color='orange',␣
 ↪label='GPU', align='center')
```

```
# Set labels and title
ax.set_xlabel('Matrix Size (M, N, K)', fontsize=12)
ax.set_ylabel('Normalized Time (0-1)', fontsize=12)
ax.set_title('Comparison of Normalized CPU and GPU Times', fontsize=14)
ax.set_xticks(matrix_sizes)
ax.set_xticklabels([f'{m},{m},{m}' for m in matrix_sizes], rotation=45)
ax.legend()

# Show gridlines
ax.grid(True, linestyle='--', alpha=0.7)

# Adjust layout for better spacing
plt.tight_layout()

# Show the plot
plt.show()
```



[37]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the new CSV data
file_path = 'gemmStaticVals.csv'
data = pd.read_csv(file_path)

# Extract the relevant columns
```

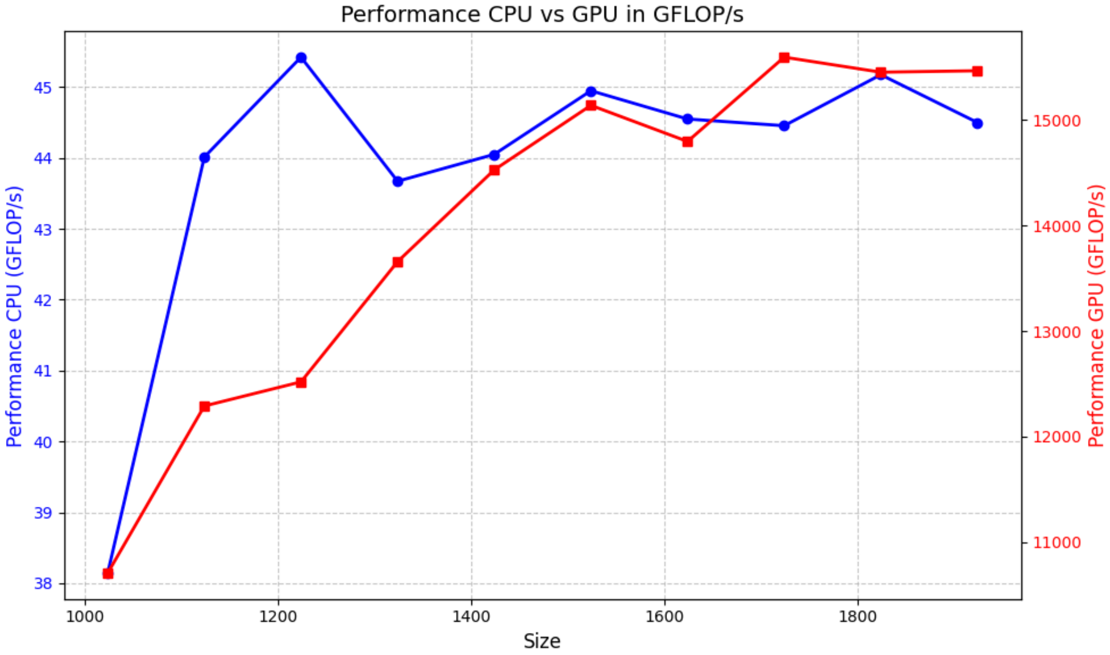
```
matrix_sizes = data['M']
cpu_times = data['AverageTimeCPU(ms)']
gpu_times = data['AverageTimeGPU(ms)']
cpu_flops = data['PerformanceCPU(GFLOP/s)']
gpu_flops = data['PerformanceGPU(GFLOP/s)']

# Create the figure and subplots
fig, ax = plt.subplots(1, 2, figsize=(14, 6))

# Plot CPU vs GPU average times on the first subplot
ax[0].plot(matrix_sizes, cpu_times, label='CPU Time (ms)', color='blue',␣
 ↪marker='o')
ax[0].plot(matrix_sizes, gpu_times, label='GPU Time (ms)', color='orange',␣
 ↪marker='o')
ax[0].set_title('Comparison of CPU and GPU Average Time', fontsize=12)
ax[0].set_xlabel('Matrix Size (M, N, K)', fontsize=10)
ax[0].set_ylabel('Average Time (ms)', fontsize=10)
ax[0].legend()

# Plot CPU vs GPU FLOPs on the second subplot
ax[1].plot(matrix_sizes, cpu_flops, label='CPU FLOPs (GFLOP/s)', color='blue',␣
 ↪marker='o')
ax[1].plot(matrix_sizes, gpu_flops, label='GPU FLOPs (GFLOP/s)',␣
 ↪color='orange', marker='o')
ax[1].set_title('Comparison of CPU and GPU FLOPs', fontsize=12)
ax[1].set_xlabel('Matrix Size (M, N, K)', fontsize=10)
ax[1].set_ylabel('FLOPs (GFLOP/s)', fontsize=10)
ax[1].legend()

# Adjust layout for better spacing
plt.tight_layout()

# Show the plot
plt.show()
```



[38]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Extraire les colonnes nécessaires
mnk_values = data['M']
cpu_performance = data['PerformanceCPU(GFLOP/s)']
gpu_performance = data['PerformanceGPU(GFLOP/s)']

# Créer la figure et les axes
fig, ax1 = plt.subplots(figsize=(10, 6))

# Premier axe Y pour la CPU
ax1.plot(mnk_values, cpu_performance, label='Performance CPU (GFLOP/s)',␣
  ↪color='blue', linewidth=2, marker='o')
ax1.set_xlabel('Size', fontsize=12)
ax1.set_ylabel('Performance CPU (GFLOP/s)', color='blue', fontsize=12)
ax1.tick_params(axis='y', labelcolor='blue')

# Deuxième axe Y pour la GPU
ax2 = ax1.twinx()  # Partage le même axe X
ax2.plot(mnk_values, gpu_performance, label='Performance GPU (GFLOP/s)',␣
  ↪color='red', linewidth=2, marker='s')
ax2.set_ylabel('Performance GPU (GFLOP/s)', color='red', fontsize=12)
ax2.tick_params(axis='y', labelcolor='red')

# Ajouter un titre et une grille
plt.title('Performance CPU vs GPU in GFLOP/s', fontsize=14)
ax1.grid(True, linestyle='--', alpha=0.7)

# Ajouter les légendes
fig.tight_layout()  # Ajuste les espacements
```

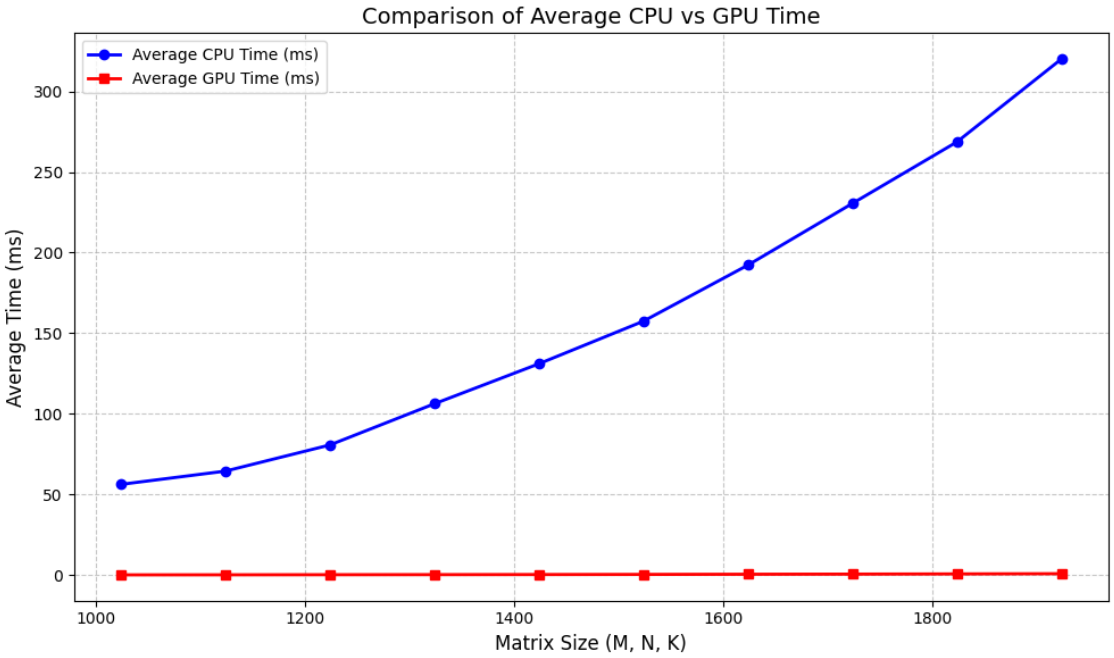
```
plt.show()
```



[40]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Extract the necessary columns
mnk_values = data['M']
cpu_times = data['AverageTimeCPU(ms)']
gpu_times = data['AverageTimeGPU(ms)']

# Create the figure and axes
plt.figure(figsize=(10, 6))

# Plot CPU and GPU times on the same scale
plt.plot(mnk_values, cpu_times, label='Average CPU Time (ms)', color='blue',␣
 ↪linewidth=2, marker='o')
plt.plot(mnk_values, gpu_times, label='Average GPU Time (ms)', color='red',␣
 ↪linewidth=2, marker='s')

# Set labels and title
plt.xlabel('Matrix Size (M, N, K)', fontsize=12)
plt.ylabel('Average Time (ms)', fontsize=12)
plt.title('Comparison of Average CPU vs GPU Time', fontsize=14)

# Show grid and legend
```

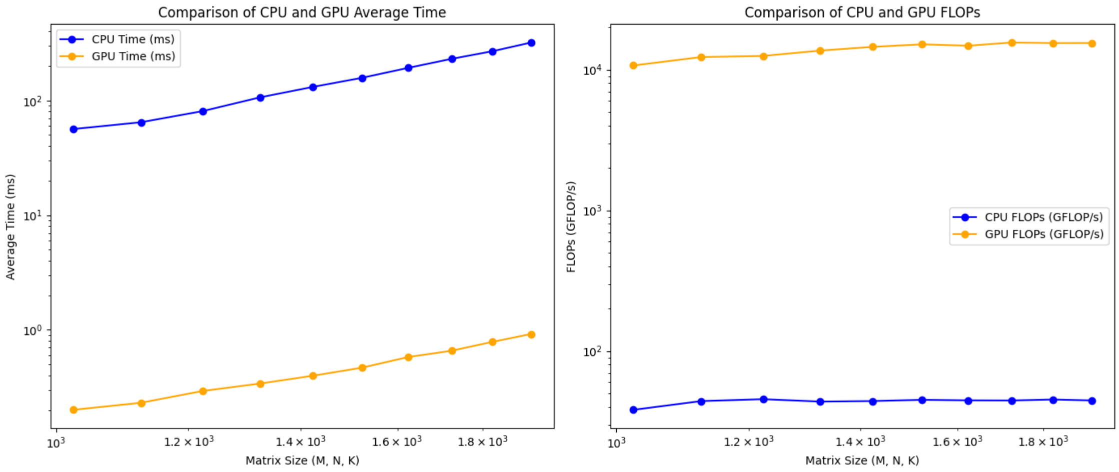
```
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend()

# Show the plot
plt.tight_layout()
plt.show()
```



[41]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the new CSV data
file_path = 'gemmStaticVals.csv'
data = pd.read_csv(file_path)

# Extract the relevant columns
matrix_sizes = data['M']
cpu_times = data['AverageTimeCPU(ms)']
gpu_times = data['AverageTimeGPU(ms)']
cpu_flops = data['PerformanceCPU(GFLOP/s)']
gpu_flops = data['PerformanceGPU(GFLOP/s)']

# Create the figure and subplots
fig, ax = plt.subplots(1, 2, figsize=(14, 6))

# Plot CPU vs GPU average times on the first subplot with log scale
```

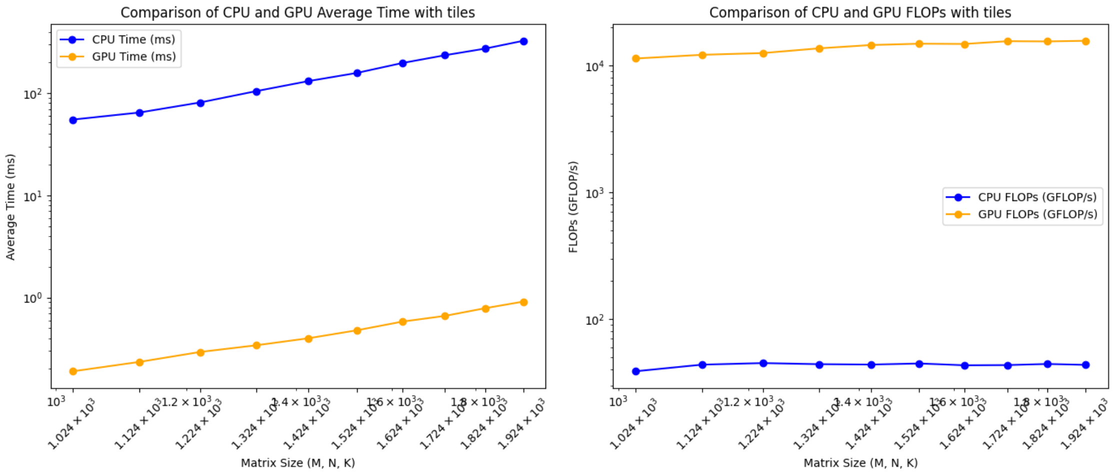
```
ax[0].plot(matrix_sizes, cpu_times, label='CPU Time (ms)', color='blue', marker='o')
ax[0].plot(matrix_sizes, gpu_times, label='GPU Time (ms)', color='orange', marker='o')
ax[0].set_title('Comparison of CPU and GPU Average Time', fontsize=12)
ax[0].set_xlabel('Matrix Size (M, N, K)', fontsize=10)
ax[0].set_ylabel('Average Time (ms)', fontsize=10)
ax[0].set_xscale('log')
ax[0].set_yscale('log')
ax[0].legend()

# Plot CPU vs GPU FLOPs on the second subplot with log scale
ax[1].plot(matrix_sizes, cpu_flops, label='CPU FLOPs (GFLOP/s)', color='blue', marker='o')
ax[1].plot(matrix_sizes, gpu_flops, label='GPU FLOPs (GFLOP/s)', color='orange', marker='o')
ax[1].set_title('Comparison of CPU and GPU FLOPs', fontsize=12)
ax[1].set_xlabel('Matrix Size (M, N, K)', fontsize=10)
ax[1].set_ylabel('FLOPs (GFLOP/s)', fontsize=10)
ax[1].set_xscale('log')
ax[1].set_yscale('log')
ax[1].legend()

# Adjust layout for better spacing
plt.tight_layout()

# Show the plot
plt.show()
```



[44]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the new CSV data
file_path = 'gemmtotask3.csv'
data = pd.read_csv(file_path)

# Extract the relevant columns
matrix_sizes = data['M']
cpu_times = data['AverageTimeCPU(ms)']
gpu_times = data['AverageTimeGPU(ms)']
cpu_flops = data['PerformanceCPU(GFLOP/s)']
gpu_flops = data['PerformanceGPU(GFLOP/s)']

# Create the figure and subplots
fig, ax = plt.subplots(1, 2, figsize=(14, 6))

# Plot CPU vs GPU average times on the first subplot with log scale
ax[0].plot(matrix_sizes, cpu_times, label='CPU Time (ms)', color='blue',␣
 ↪marker='o')
ax[0].plot(matrix_sizes, gpu_times, label='GPU Time (ms)', color='orange',␣
 ↪marker='o')
ax[0].set_title('Comparison of CPU and GPU Average Time with tiles',␣
 ↪fontsize=12)
ax[0].set_xlabel('Matrix Size (M, N, K)', fontsize=10)
ax[0].set_ylabel('Average Time (ms)', fontsize=10)
ax[0].set_xscale('log')
ax[0].set_yscale('log')
ax[0].set_xticks(matrix_sizes)  # Ensure proper scaling on x-axis
ax[0].tick_params(axis='x', rotation=45)  # Rotate x-axis labels by 45 degrees
ax[0].legend()

# Plot CPU vs GPU FLOPs on the second subplot with log scale
ax[1].plot(matrix_sizes, cpu_flops, label='CPU FLOPs (GFLOP/s)', color='blue',␣
 ↪marker='o')
ax[1].plot(matrix_sizes, gpu_flops, label='GPU FLOPs (GFLOP/s)',␣
 ↪color='orange', marker='o')
ax[1].set_title('Comparison of CPU and GPU FLOPs with tiles', fontsize=12)
ax[1].set_xlabel('Matrix Size (M, N, K)', fontsize=10)
ax[1].set_ylabel('FLOPs (GFLOP/s)', fontsize=10)
ax[1].set_xscale('log')
ax[1].set_yscale('log')
ax[1].set_xticks(matrix_sizes)  # Ensure proper scaling on x-axis
ax[1].tick_params(axis='x', rotation=45)  # Rotate x-axis labels by 45 degrees
ax[1].legend()

# Adjust layout for better spacing
```

```
plt.tight_layout()

# Show the plot
plt.show()
```



[ ]: